Node activity scheduling in wireless sensor networks

Saoucene Mahfoudh, Pascale Minet

In Proc. of the 17th International Conference on Real-Time and Network Systems
RTNS'2009, Paris, ECE, 26-27 October, 2009

Node activity scheduling in wireless sensor networks*

Saoucene Mahfoudh and Pascale Minet
INRIA
Rocquencourt
78153 Le Chesnay cedex, France
saoucene.mahfoudh@inria.fr, pascale.minet@inria.fr

Abstract

Wireless sensor networks have resources of limited capacity (e.g. bandwidth, processing power, memory and energy). That is why these resources should be efficiently used. Node activity scheduling is a technique that allows nodes to alternate sleep and awake states. This technique spares energy insofar as the sleep state is the state using the smallest power. Moreover, by allowing several nodes to transmit simultaneously without interfering, spatial reuse of the bandwidth is obtained. Furthermore with a smart schedule, data gathering can be done in a single cycle. All these reasons render node activity scheduling very attractive in wireless sensor networks. We propose in this paper a three-hop coloring algorithm for data gathering applications. Simulation results allow us to evaluate the number of colors needed to color all network nodes and hence to determine the reduced size of the activity period in each polling cycle. The complexity of our algorithm is given in terms of number of messages sent per node. We can then determine the network configurations for which coloring brings interesting benefits, namely a more efficient use of the bandwidth and the node energy, as well as a shorter delay to collect data ensuring their time consistency.

1. Introduction

With the increasing number of applications in domains as various as environment protection (detection of forest fire or seismic event, wild life protection), civilian protection (building and bridge monitoring), emergency rescue, exploration mission in hostile environments and home monitoring, wireless ad hoc and sensor networks have a promising future. However, nodes in such networks can have a limited amount of energy. This energy can be difficult, expensive or even impossible to renew. The protocols operating in such networks should be energy efficient to maximize the network lifetime. Node activity scheduling is one class of energy efficient techniques [1]. Since the sleep state is the least power consuming state, compared with the receive, transmit and idle states, the idea of the node activity scheduling protocols is to schedule node state between sleeping and active to minimize energy consumption while ensuring network and application functionalities.

Node activity scheduling allows nodes not only to spare energy but also to use bandwidth more efficiently. Some solutions take advantage of spatial reuse to determine the time intervals dedicated to node activity. Indeed, during the same time interval two transmitters can transmit simultaneously and successfully if they do not interfere. Spatial reuse can be obtained by means of a coloring algorithm. The coloring algorithm must allow unicast as well as broadcast transmissions. For instance, nodes need to broadcast $Hello$ messages in their one-hop neighborhood to indicate that they are still alive and to declare on the one hand the nodes they hear only (i.e. unidirectional links), and on the other hand the nodes they hear and are heard from (i.e. bidirectional links). Moreover as the radio propagation is versatile, an immediate acknowledgement is often required in unicast transmissions. Thus the sender is ensured that the (one-hop) receiver has correctly received its message, otherwise it retransmits it.

In this paper, we focus on node activity scheduling obtained with a node coloring algorithm and study the benefits brought to a data gathering application. The collected data correspond to a physical phenomenon that evolves with time. It is then essential to minimize the delay needed to collect these data from sensors generating them. Furthermore, ensuring that all data are gathered in a single polling cycle guarantees their time consistency. To achieve these goals and then maximize the advantage brought by coloring, coloring must take into account the data gathering tree in color selection. Otherwise, it can take, in the worst case, a number of polling cycles equal to the number of hops to reach the data sink from the considered sensor. The data gathering tree can be computed by the Prim's algorithm where the sink broadcasts a message to all sensor nodes. This message contains a number representing the level of the sending node in the tree. Initially, this number is set to 0 by the sink and is incremented at each retransmission.

This paper deals with node activity scheduling in wireless sensor networks used by data gathering applications. It is

*This study has been partly funded by the ANR OCARI project.

organized as follows. In Section 2 we give a brief state of the art related to node activity scheduling distinguishing four classes of solutions. We then recall the complexity of node coloring and present different types of coloring algorithms. In section 3, we first justify the design choices of our algorithm. The main principles of the coloring algorithm are given as well as the messages exchanged and the information maintained by each node. The algorithm itself is described. The performances of this algorithm are evaluated in Section 4 by means of simulations with different network configurations. The number of colors allows us to determine when coloring provides real benefits. The average number of messages sent per node is an indication of the induced overhead. With this information, we can then evaluate the activity period and the data gathering delay. In Section 5, we show how the coloring algorithm can adapt to message losses, tree changes and late arrivals of nodes. Finally, we conclude in Section 6.

2. State of the art

We first present different techniques for scheduling node activity. We then move to graph coloring that can be used to schedule node activity.

2.1. Node activity scheduling

The best way to save energy nodes is to turn off the sensor radio when it does not receive or transmit data, so keeping sensor nodes in the sleep state. This must be accompanied by a node scheduling activity to prevent network partition and message loss when some nodes are in sleep state. We can distinguish four classes of node activity scheduling

- Computation of connected sets of active nodes. In [2], Simplot et al propose a solution building a connected dominating set (i.e. each node is either in this subset or is a neighbor of a node in this subset). Only the nodes of this set are active. All other nodes can change their state to sleep mode. It is a distributed and localized solution: only information about one hop and two hops neighbors is needed. Other solutions like [3] propose to extend network lifetime by dividing the network nodes in disjoint sets, such that each node set meets the network and application functions. These sets are activated successively, and at any time only the nodes of one set are active. All others nodes are in the sleep state. To improve network lifetime, the number of disjoint sets must be maximized which is NP-complete. The solution proposed is centralized. These authors have shown in [4] that network lifetime can be improved by allowing non-disjoint sets.

- CSMA/CA. In [5], S-MAC, an energy efficient MAC protocol for sensor networks is introduced. The main goal of S-MAC protocol is to reduce energy consumption by using a periodic sleep-wake up cycle, while supporting good scalability and collision avoidance. It consists of three major components: 1) periodic listen and sleep, 2) collision and overhearing avoidance, and 3) message passing. It is based on a CSMA/CA channel access and the RTS/CTS mechanism, whose overhead reduces the protocol efficiency. Many other variations of S-MAC are proposed like T-MAC [6] with an adaptive length of active state, D-MAC [7] to reduce the network latency, O-MAC [8] to improve the throughput...

- TDMA. In its basic version, TDMA provides one transmission slot per node in the network. It provides a guaranteed access per cycle for every node and avoids collisions. However, it does not adapt to traffic variations. Many improvements have been proposed in order to take advantage of spatial reuse in wireless networks. Among them, USAP (Unifying Slot Assignment Protocol) [9, 10] has drawn a lot of attention. USAP [9] is a distributed TDMA slot assignment protocol for mobile multihop packet radio networks. It allows any node N_i to select a slot for transmission that is unassigned in its neighborhood. A slot is unassigned from N_i point's of view if no one-hop neighbor of N_i transmits or receives during this slot.

  In [11], a deterministic solution based on slot assignment named TRAMA is proposed. It consists in three modules: 1) a neighborhood discovery protocol, 2) a schedule exchange protocol and 3) an adaptive election algorithm that selects the transmitter and receiver(s) for each time slot. Only nodes having data to send contend for a slot; notice however, that a node does not know which of its 1-hop and 2-hop neighbors have data to send. The node with the highest priority in its two-hop neighborhood wins the right to transmit in the slot considered. Each node declares in advance its next schedule containing the list of its slots and for each slot its receiver(s). The adaptivity of TRAMA to the traffic rate comes at a price: its complexity. To reduce the complexity of TRAMA, FLAMA [12] is introduced. However, this protocol is designed only for data gathering applications in sensor networks based on tree structure. The protocol is simplified both in terms of message exchange and processing complexity. The number of slots allocated by FLAMA to a node with a given traffic rate highly depends on node priority computation.

- Hybrid. Z-MAC [13] operates like CSMA under low contention and like TDMA under heavy contention, reducing collisions among two-hop neighbors by means of an initial slot assignment made by DRAND [14]. The goal of Z-MAC is to optimize the bandwidth efficiency of the MAC protocol, selecting CSMA/CA and TDMA when they exhibit the best performance. We can notice that Z-MAC does not allow an immediate acknowledgement of unicast messages. Indeed, this acknowledgement could cause a

conflict, as illustrated in Figure 1. Moreover, since a slot that is unused by its owner can be used by one of its neighbors, the nodes must stay awake in order to be able to receive this message if they are the destination. From the energy point of view, Z-MAC reduces the activity period in the polling cycle enforced by the application but does not allow nodes to sleep during the activity period, what does our coloring algorithm presented in Section 3, whose goal is to maximize network lifetime by scheduling node activity. DRAND [14] assigns slots to nodes in such a way that one-hop and two-hop neighbors have different slots. This randomized algorithm has the advantage of not depending on the number of network nodes but at the cost of an asymptotic convergence.

It appears that different techniques have been used to schedule node activity. The techniques based on CSMA variants behave well in case of light loads and easily adapt to topology changes, whereas techniques based on TDMA variants outperform them in case of high loads. In this study, we focus on wireless sensor nodes, where only a few nodes are mobile. Graph coloring has been introduced to increase the efficiency of TDMA with spatial reuse: two nodes with the same color transmit simultaneously without interfering [15, 23].

2.2. Graph coloring

One-hop graph coloring consists in coloring nodes with the minimum number of colors such that two adjacent nodes have not the same color. The problem of one-hop coloring has been shown NP-complete in [18] for the general case. The first one-hop graph coloring algorithms proposed were centralized. Among the greedy algorithms (i.e. no color backtracking), Dsatur, [16, 17], where the vertex with the highest number of already colored neighbor vertices is colored first, exhibits very good performances, even if it is not optimal. It is then followed by Largest First, where the node with the highest degree is colored first.

Distributed one-hop graph coloring algorithms also exist. Some are probabilistic such as [20, 19]. Other algorithms are deterministic such as [22] where Distributed Largest First (DLF) is proposed. Another approach consists in finding maximum independent sets and then coloring these sets independently, as in [24], because both problems are related [21].

The efficiency of a distributed coloring algorithm, [22, 21], can be evaluated by:

- the number of colors needed to color a graph G.
- the average number of messages sent per node. This shows the overhead induced by the coloring algorithm.
- its time complexity, expressed in the case of a distributed algorithm, by the maximum number of rounds needed to color each node. A round is such that every node can:
  - send a message to all its one-hop neighbors,
  - receive the messages sent by them,
  - perform some local computation based on the information contained in the received messages.

In wireless ad hoc and sensor networks, distributed coloring algorithms based on decision rounds require less messages than those based on the alternation of proposal/decision rounds, such as Distributed Largest First [22]. A comparative performance evaluation can be found in [15] for two-hop graph coloring. In fact, in a round, a node sends a message to its one-hop neighbors, this message contains its color and the colors of its one-hop neighbors. It receives the messages from its one-hop neighbors and takes a decision if it has the highest priority. This is the principle of our coloring algorithm presented in Section 3.
Two-hop graph coloring has also been investigated in the case of data gathering applications [23], where the goal is to reduce the duration needed to gather data from all sensors. Starting to color the highest level in the tree and ending by the sink as in [23] is easy, but the insertion of a new leaf can lead to recolor the tree.

3. Algorithm for three-hop coloring of a tree and properties

3.1. Design choices

From the state of the art, it turns out that:

- the coloring problem being NP-complete, we seek for an heuristic. The state of the art points out that the algorithms achieving the best performances are those taking into account the node degree (i.e. number of one-hop neighbors if one-hop coloring) and having identical rounds (no alternate of proposal/decision rounds).
- Furthermore, link coloring specifies the sender and the receiver whereas node coloring specifies only the sender. Hence, only node coloring enables broadcast transmissions.
- Node coloring must be such that two nodes having the same color can transmit simultaneously without interfering. Interferences being assumed to be limited to two-hop, two-hop coloring is needed. However, as Figure 1 shows, it is not sufficient if a node is allowed to transmit an immediate acknowledgement of the received data: nodes A and D, 3-hop away, use the same color and the $data$ message sent by A collides with the ack message sent by C to node D. Hence, three-hop coloring is needed to accommodate immediate acknowledgement of unicast

transmissions. In other words, with three-hop coloring, the same color can be reused four-hop away. For instance, nodes A and E can have the same color. In such a case, node A can transmit data to B while D is acknowledging the data received from E without interfering.

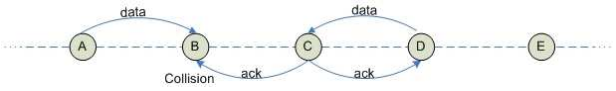

Figure 1. Collision with two-hop coloring and immediate acknowledgement.

- Without care, node activity scheduling increases the delay needed to collect information from a sensor. In the worst case, a number of cycles equal to the distance, in hop number, to the sink is needed to reach the sink. To avoid this phenomenon, a node must transmit its data before its parent in the data gathering tree. To achieve that, the color of a node must be higher than the color of its parent in the tree and slots are assigned in the decreasing order of colors.

3.2. Notations

We consider any data gathering tree $\mathcal{T}$ and any node N in this tree.
Let $\mathcal{N}^3(N)$ denote the neighborhood up to 3 hops from N. This set contains the 1-hop, 2-hop and 3-hop neighbors of N.
Let $parent(N)$ denote the parent of N in $\mathcal{T}$. By descendant, we mean the children, children of the children and so on...
Let $Desc(N)$ denote the set of descendants of N in $\mathcal{T}$.
Let $|Desc(N)|$ denote the cardinal of this set.

For any node N, we say that N has a higher priority than node $N' \in \mathcal{N}^3(N)$ if and only if:

- either N' meets $|Desc(N')| > |Desc(N)|$
- or $(|Desc(N')| = |Desc(N)|$ and $identifier(N') < identifier(N))$.

Colors are assumed to be represented by positive integers starting with 0, the color of the root of the data gathering tree.

3.3. Principles

To achieve the goals given in section 1, our coloring algorithm is based on the two following rules:

- **Rule R1:** any node N colors itself if and only if:
  - all nodes in $\mathcal{N}^3(N)$ having a strictly higher number of descendants are already colored,
  - and all nodes in $\mathcal{N}^3(N)$ having the same number of descendants and a smaller identifier are already colored.
- **Rule R2:** node N takes the smallest color available in $\mathcal{N}^3(N)$ strictly higher than the color used by its parent.

It follows that:

- the first node to color is the root of the tree.
- each node has a color strictly higher than the color of its parent and of all its ascendants.
- each node has a color higher than or equal to its level in the tree (i.e. distance to the root in the tree expressed in hop number).

Figure 2 depicts the colors assigned to 15 nodes called A,B...O building a tree rooted at node A. Eight colors are needed. Hence, eight slots are sufficient instead of 15 for classical TDMA. The same color can be used by several nodes (e.g; color 6 is used by nodes K, L, M and O).

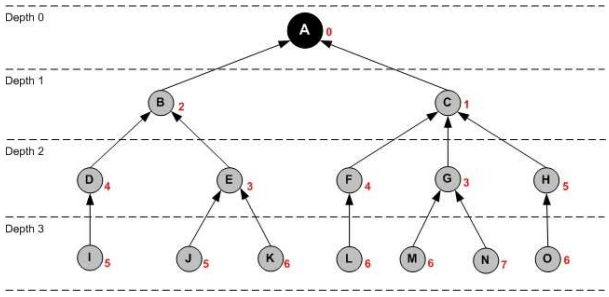

Figure 2. Colors assigned to a tree of 15 nodes.

Figure 3 depicts the slot assignment. For instance, nodes K , L, M and O that have the color 6 transmit in the same slot, numbered 6. Notice that each child transmits before its parent. Hence, in the cycle of eight slots illustrated by this figure, any sensor can send its value to the sink, assuming data aggregation.

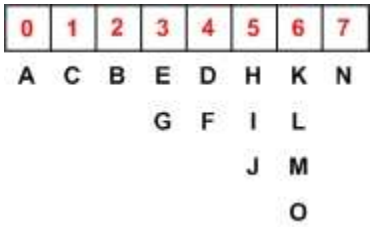

Figure 3. Slots assignment.

3.4. Messages exchanged

In this algorithm, two messages are exchanged:

- the $Color$ message: it is the only message needed to color all network nodes.
- the $MaxColor$ message: this message is needed at the end of the coloring algorithm to inform the root of the tree of the number of colors used in the tree.

The $Color$ message contains:

- the node identifier,

- its sequence number,
- its number of descendants,
- its color, if already assigned,
- for any one-hop neighbor node:
  - the node identifier,
  - its sequence number,
  - its number of descendants,
  - its color, if already assigned,
- for any two-hop neighbor node:
  - the node identifier,
  - its number of descendants,
  - its color, if already assigned,

The $Color$ message is broadcast one-hop. The sequence number of the sender is incremented at each change in the fields of the $Color$ message, except the sequence number fields. In order to tolerate message losses, a node N retransmits its $Color$ messages until all its one-hop neighbors have sent a $Color$ message containing a sequence number for N equal to the current one. Notice that in case of topology changes, a link can be broken causing an update in the set $\mathcal{N}^3(N)$.

The $Maxcolor$ message contains:

- the node identifier of the sender,
- its sequence number,
- the maximum color used by all the descendants of the sender node.

The $MaxColor$ message is sent from any node N to its parent until reaching the root. This unicast message is acknowledged. When it reaches the root, it contains the maximum number of colors used.

3.5. Information maintained by each node

Each node maintains the following information:

- its node identifier,
- its sequence number,
- its number of descendants,
- its color, if already assigned,
- its parent in $\mathcal{T}$,
- for each child in $\mathcal{T}$,
  - the node identifier,
  - the maximum color used by this node and its descendants,
  - a sequence number,
- for any one-hop neighbor node:
  - the node identifier,
  - its sequence number,
  - its number of descendants,
  - its color, if already assigned,
- for any two-hop neighbor node:
  - the node identifier,
  - its number of descendants,
  - its color, if already assigned,
- for any three-hop neighbor node:
  - the node identifier,
  - its number of descendants,
  - its color, if already assigned,

3.6. Algorithm

We now give the three-hop coloring algorithm for a data gathering application:

Algorithm 1 Three-hop coloring algorithm of a data gathering tree

1: **Procedure Process(**$Color$**message)**
2: **begin procedure**
3: **if** there is a change in the 1, 2 or 3-hop neighborhood or in the tree **then**
4: update $\mathcal{N}^3(N)$
5: update $\mathcal{T}$, $parent(N)$ and $\mathcal{D}esc(N)$
6: **end if**
7: **if** $|Desc(N)|$ has not yet been computed and the 1,2 and 3-hop neighborhoods as well as $|Desc(M)|$ for each child M are known **then**
8: compute $|Desc(N)|$ by adding the values of $1 + |Desc(M)|$ for each M, child of N.
9: **end if**
10: maintain the priority and color of any node in $\mathcal{N}^3(N)$
11: **if** N is the node with the highest priority among the uncolored nodes in $\mathcal{N}^3(N)$ **then**
12: N selects the smallest color unused in $\mathcal{N}^3(N)$ strictly higher than $color(parent(N))$
13: **end if**
14: **end procedure**
15: **Main**
16: **repeat**
17: **repeat**
18: N broadcasts (1-hop) its $Color$ message containing:
- a sequence number seq incremented at each change in the $Color$ message fields, except the sequence number fields
- its identifier
- its priority and color if already assigned
- its list of one-hop neighbors with their identifier, their sequence number, their priority and color if already assigned
- its list of two-hop neighbors with their identifier, their priority and color if already assigned.
19: N waits for the $Color$ message of its 1-hop neighbors
20: **upon receipt of a** $Color$ **message,**
21: N Process($Color$message)
22: **until** all one-hop neighbors have received the $Color$ message of N with number seq
23: **until** all nodes in $\mathcal{N}^3(N)$ are colored

3.7. Computation of the number of colors

We now evaluate the number of colors needed to color all network nodes. We first assume that the neighborhood up to 3-hop does not bring additional constraints to the one represented in the tree.

Property 1: If at each level p in the tree, each node has the same number of children $nbchild_p$ and there is no additional constraints to the one depicted in the tree, then the number of colors used is equal to:

$$nbcolor = 1 + \sum_{p=0}^{depth-1} nbchild_p.$$

If $\mathcal{N}^3(N)$ introduces additional constraints to those given in the tree, there exists a node N that has for 1-hop, 2-hop or 3-hop a node N' that is neither its parent, grandparent, brother, uncle, child, nephew, grandchild. in such a case, colors cannot be reused so easily, we then get:

Property 2: If at each level p in the tree, each node has the same number of children $nbchild_p$ and there are additional constraints to the one depicted in the tree, then the number of colors used is equal to:

$nbcolor = 1 + \sum_{p=0}^{depth-1} nbchild_p + \sum_{N} \delta_{N,N'}$, with $\delta_{N,N'} = 1$ if and only if N cannot take a color in $[color(parent(N)) + 1, cmax]$ with $cmax$ the highest color used by the nodes N' colored before N. Initially $cmax = color(parent(N)) + 1$ and $cmax = cmax + 1$ each time a new color is used to color N.

We can get the formula giving the color of a node, depending on the color of its parent and the colors already used in its neighborhood up to three hops.

Property 3: The color of any node N is given by:

$color(N) = 1 + color(parent(N)) + \sum_{N' \ already \ colored \in \mathcal{N}^3(N)} \delta_{N,N'}$

with $\delta_{N,N'} = 1$ if and only if N cannot take a color in $[color(parent(N)) + 1, cmax]$ with $cmax$ the highest color used by the nodes $N' \in \mathcal{N}^3(N)$ colored before N. Initially $cmax = color(parent(N)) + 1$ and $cmax = cmax + 1$ each time a color in $[color(parent(N))+1, cmax]$ cannot be used to color N.

3.8. Comparison with another tree coloring algorithm

In this section, we compare our coloring algorithm with another one proceeding level by level (from the lowest level to the highest one) and applying the rules R'1 and R2 instead of rules R1 and R2, where the rule R'1 is:

- **Rule R'1:** any node N colors itself if and only if:
  - any node N' in $\mathcal{N}^3(N)$ having a level strictly lower than the level of N is already colored,
  - and any node N' in $\mathcal{N}^3(N)$ of the same level as N but $|\mathcal{N}^3(N')| > |\mathcal{N}^3(N)|$ is already colored,
  - and any node N' in $\mathcal{N}^3(N)$ of the same level as N but $|\mathcal{N}^3(N')| = |\mathcal{N}^3(N)|$ and a smaller identifier than N is already colored.

- **Rule R2:** node N takes the smallest color available in $\mathcal{N}^3(N)$ strictly higher than the color used by its parent.

The simple example depicted on Figures 4 and 5 shows that the number of colors used by our algorithm (here 4 colors) is smaller than this used by this other algorithm (here 5 colors). The reason comes from the fact that our algorithm favors the nodes with a high number of descendants. As they are colored first, they can get smaller colors unlike in the other algorithm. The nodes with a small number of descendants are colored at the end, but they can reuse colors used by nodes that are not in their up to 3-hop neighborhood. We will see in subsection 4.4 that the number of colors used by our algorithm is considerably smaller than the number of colors used by the other algorithm using rules R'1 and R2.

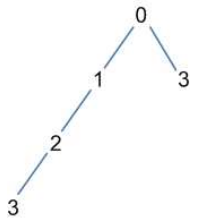

Figure 4. Tree colored with our algorithm.

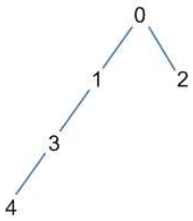

Figure 5. Tree colored with the other algorithm.

4. Performance evaluation

4.1. Simulation parameters

For performance evaluation purpose, we have considered different network configurations. Each configuration is characterized by a node number and a node density. The nodes are randomly deployed over the network area. The number of nodes ranges from 50 to 200, whereas the node density (i.e. the average number of one-hop neighbors per node) ranges from 10 to 20. For each configuration, we have run 10 simulations and the result depicted in the curves is the average of these 10 simulations. The data gathering tree is the tree of the shortest paths. This study can also be extended to the case where the data gathering tree is built taking into account residual energy of nodes and their capacity to forward messages.

For all the network configurations generated, we have computed different performance criteria such as the number of colors needed to color all network nodes, the average number of messages sent per node. We then show how this small number of colors benefits to the active period duration and the data gathering delay.

4.2. Number of colors

We now evaluate the number of colors needed by our algorithm to color all network nodes of the generated configurations. Simulation results are illustrated in Figures 6 and 7.

We first depict in Figure 6 the number of colors as a function of the number of nodes for different densities. We observe that this number strongly depends on node density and weakly on node number. As the color a node N cannot be reused in its neighborhood up to 3-hop, whose size is proportional to the density, this explains the strong dependency between the number of colors and the density. It is very interesting to notice that all curves depicting the number of colors are below the first bisectrix. This means that the number of colors is much smaller than the number of nodes, except for the network configuration of 50 nodes and a density of 20, where almost all nodes are 1, 2 or 3-hop neighbors. In other words, spatial reuse is always obtained for all the network configurations considered in the simulations. The higher the distance to the first bisectrix, the higher the spatial reuse. The average number of nodes using the same color is given by the number of nodes divided by the number of colors.

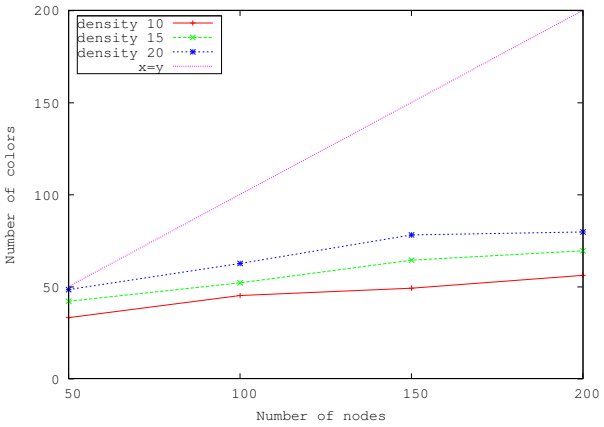

Figure 6. Number of colors as a function of the number of nodes and the density.

We now depict in Figure 7 the number of colors as a function of the tree depth for different densities. Notice that for a given density, the number of nodes increases with the tree depth. The number of colors increases with the tree depth, this is due to the fact that the color of a node is higher than the color of its parent. Moreover, the impact of tree depth seems to stabilize in the simulated configurations. The stabilization point is reached sooner for high densities.

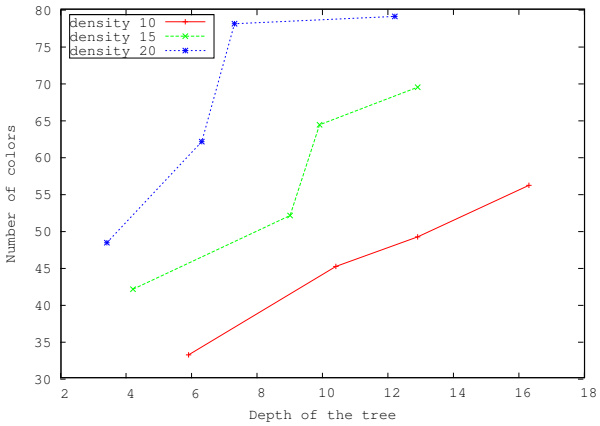

Figure 7. Number of colors as a function of the tree depth and the density.

4.3. Average number of messages sent by a node

In order to evaluate the overhead induced by the coloring algorithm, we compute the average number of messages sent per node in all the generated network configurations. Simulation results are illustrated in Figure 8. The number of messages exchanged strongly increases with density and moderately with the number of nodes. It remains reasonable in all configurations.

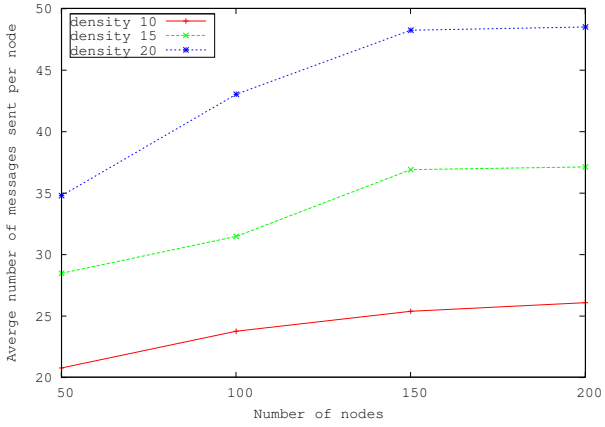

Figure 8. Average number of messages sent per node.

4.4. Comparative evaluation

We now compare the number of colors needed in our coloring algorithm with this needed by the other algorithm presented in subsection 3.8. We consider a network density of 10. Our algorithm exhibits very good performances as illustrated in Figure 9. This is due to the fact that it assigns a smaller color to nodes having a high number of descendants. Other nodes (with a small number of descendants) can reuse already used colors. Hence, the smallest number of colors. We can also observe that this trend increases with the number of nodes.

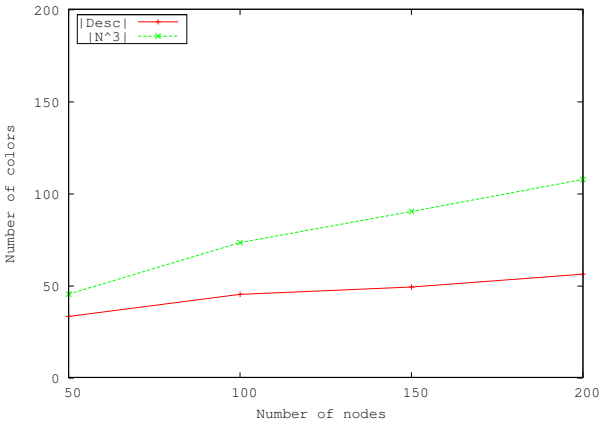

Figure 9. Comparative evaluation of the number of colors used by the $|Desc(N)|$ and $|\mathcal{N}^3(N)|$ algorithms.

4.5. Activity period duration

The physical phenomenon that is monitored by the wireless sensor network determines the polling cycle of the sensors. This polling cycle comprises two periods, as represented in Figure 10: an activity period during which sensors are active and an inactive period where all sensors are sleeping. Only the activity period is mandatory.

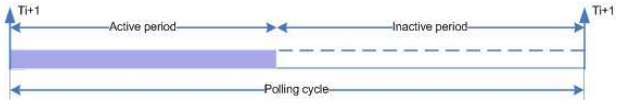

Figure 10. The active period in the polling cycle.

The coloring algorithm is used for slot assignment in the activity period. Slots are assigned to nodes per color instead per node like in classical TDMA: all nodes of the same color can transmit simultaneously without interfering. Hence, this space reuse leads to a considerable reduction of the activity period. The benefit is equal to $(numberofnodes - numberofcolors) \cdot slotsize$.

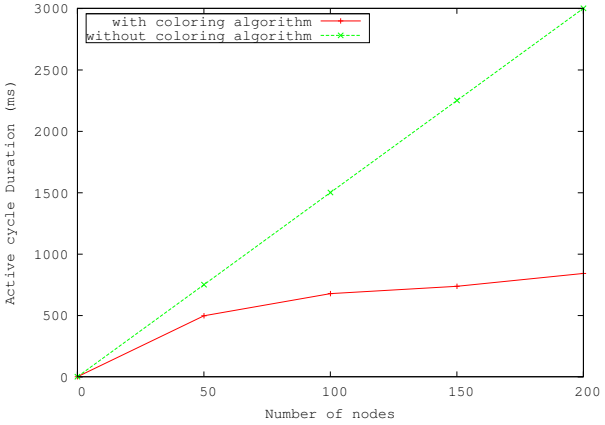

Figure 11. Duration of the active period as a function of the number of nodes.

Figure 11 shows the quantitative improvement brought by the coloring algorithm with regard to a classical TDMA. A slot size of 15 ms has been taken.

4.6. Data gathering delay

The data gathering delay is the time needed to collect at the sink all data transmitted by the sensors. If we assume that the slot has the capacity to contain the aggregated information transmitted by any child of the root (i.e.: the nodes having the highest quantity of information to transmit), then the time needed to collect the data from all sensors is equal to the activity period, whereas the latency is equal to the polling cycle duration.
Ensuring that each node has a color higher than its parent in the tree allows our colring algorithm to reduce the data gathering delay and to obtain time-consistent data. In fact, in only one cycle, all data collected can reach the sink by assigning slots to colors in the decreasing order: this allows a child to transmit before its parent that can then aggregate the data received from its children. The increasing order is chosen in reverse case of data dissemination. Moreover, collecting data in a single cycle guarantees time consistency of the data collected from the different sensors: for example temperature, humidity degree and pression are measured in the same cycle.

4.7. Benefit brought by coloring

A coloring algorithm brings several advantages:

- an efficient use of the bandwidth by enabling spatial reuse and avoiding medium access contention.
- an increased network lifetime by enabling nodes to sleep during the activity period and the increased inactivity period. During the activity period, a node sleeps in all the slots except:
  - the slot assigned to its color to transmit its messages,
  - the slots assigned to its one-hop neighbors to receive their messages in case it is the destination.

  Optimizations can be brought in order to allow a node to sleep the soonest possible. For example, if a node has not detected any signal in the slot of its one-hop neighbor during a predetermined duration, it can deduce that this neighbor has nothing to transmit and goes back to the sleep state. Similarly, a node can sort the messages it has to transmit: firts the broadcast transmissions and second the unicast transmissions ordered by the destination identifier. It follows that a one-hop neighbor can easily detect that there is no message for it and then sleeps again.
- a shorter delay to collect data from sensors. In a single cycle, all data can be collected, ensuring their time consistency.

- slots assigned according to the increasing order of colors reduces the time needed to disseminate data in the network: for example, information sent by the sink to all nodes.

5. Adaptivity of the coloring algorithm

5.1. Message loss

Our coloring algorithm tolerates message losses by means of the sequence number used to detect losses. A node N retransmits its $Color$ message with sequence number seq as long as it has not received a $Color$ message from all its one-hop neighbors containing the sequence number seq for N.

When a one-hop neighbor N' receives the $Color$ message of N, it updates the neighborhood information locally stored. Two cases are possible:

- There is a change concerning:
  - either the color or the number of descendants of the node itself, or of a one-hop or two-hop neighbor,
  - or the appearance or disappearance of a one-hop or two-hop neighbor.

  N' increments its own sequence number seq' and sends a $Color$ message to its one-hop neighbors.
- Otherwise N' sends a $Color$ message without incrementing its sequence number.

It follows that any node will know a change in its neighorhood up to 3-hop, whatever the change is: node, color or number of descendants.

5.2. Tree change

In this section, we consider the case where the link between a node N and its parent in the tree is broken. This node N will first try to attach itself to another node N' in the tree. We assume that this link is an existing link: the parent of N is chosen among its existing one-hop neighbors. Hence this parent is already taken into account in $\mathcal{N}^3(N)$. We will see in the next section what happens when this assumption is not true.

Two cases are possible:

- the tree change does not impact the colors already assigned: no node in $\mathcal{N}^3(N)$ has the same color as N and $color(N) > color(N')$.
- the tree change creates a violation of the rule 2: the color of N is not higher than the color of its new parent N'. In this case, node N selects the smallest color available in $\mathcal{N}^3(N)$ belonging to $[color(N') + 1, min_{child} color(child) - 1]$. If there is no color available, N takes the smallest color of its children, and so on.

5.3. Topology changes

By topology change, we mean that a change in $\mathcal{N}^3(N)$ occurs: a new link is created or an existing link is broken. It can be caused by:

- node mobility: a node moves in the network area causing the breakage of its existing links and the creation of new ones.
- late node arrival: when a new node joins the already colored network, new links will appear.

In both cases, when a new link is created, it may have as consequence that two nodes that were not 1-hop, 2-hop or 3-hop neighbors become 1-hop, 2-hop or 3-hop neighbors. Hence, a color conflict between two nodes can occur. A color conflict is defined as follows:

A color conflict occurs between two nodes having the same color when these nodes prevent each other or some neighbor destination to receive correctly the intended message because of a collision.

Notice that in the absence of mobility and late node arrival, nodes that are 1, 2 or 3-hop neighbors never conflict by definition of the algorithm. Furthermore, this definition takes advantage of the capture effect and considers the only color conflicts where the intended destination is prevented to receive its message.

When a color conflict occurs between two nodes N and N', the node with the highest priority keeps its color whereas the node with the smallest priority takes another color according to rule R2.

6. Conclusion

In this paper, we proposed a three-hop coloring algorithm for wireless sensor networks supporting data gathering applications. We shown that this algorithm uses a small number of colors despite the rule that the color of a node is higher than the color of its parent in the data gathering tree. This algorithm has been designed for the ANR OCARI project [25] for a medium whose physical layer is the IEEE 802.15.4. The benefits brought by this coloring algorithm are quadruple:

- the bandwidth is used more efficiently, taking advantage of spatial reuse and avoiding medium access contention by means of colors;
- the nodes spare their residual energy in sleeping while they have no message to send or to receive;
- the data transmitted by the sensors can be collected in a single activity period, ensuring their time consistency.
- colors can also be used to disseminate information from the sink to all sensor nodes in a single cycle.

Our coloring algorithm contributes to maximize network lifetime by reducing the activity period as well as allowing any node to sleep in this period (in the slots that are not assigned to its color and the colors of its 1-hop neighbors).

References

[1] S. Mahfoudh, P. Minet, *Survey of energy efficient strategies in wireless ad hoc and sensor networks*, ICN 2008, IEEE International Conference on Networking, Cancun, Mexico, April 2008.

[2] J. Carle, D. Simplot-Ryl, *Energy-Efficient Area Monitoring for Sensor Networks*, Computer, vol. 37, no. 2, pp. 40-46, Febuary, 2004.

[3] M. Cardei, M. Thai, Y. Li, W. Wu, *Energy-efficient target coverage in wireless sensor networks*, IEEE INFOCOM'05, Miami, Florida, March 2005.

[4] M. Cardei, D. Du, *Improving wireless sensor network lifetime through power aware organization*, ACM Jal of Wireless Networks, May 2005.

[5] W. Ye, J. Heidmann, D. Estrin, *An Energy-Efficient MAC Protocol for Wireless Sensor Networks*, IEEE INFOCOM, New York, USA, June 2002.

[6] T. V. Dam, K. Langendoen, *An adaptive energy-efficient MAC protocol for wireless sensor networks*, ACM SenSys'03, November 2003.

[7] G. Lu, B. Krishnamachari, C. Raghavendra, *An Adaptive Energy-Efficient and Low-Latency MAC for Data Gathering in Wireless Sensor Networks*, Parallel and Distributed Processing Symposium, April 2004.

[8] G. Lu, B. Krishnamachari, C. Raghavendra *O-MAC: An Organized Energy-Aware MAC Protocol for Wireless Sensor Networks*, IEEE ICC, Glasgow, UK, June 2007.

[9] C.D. Young, *USAP: a unifying dynamic distributed multichannel TDMA slot assignment protocol* IEEE MILCOM 1996, Vol. 1, October 1996.

[10] C.D. Young, *USAP multiple access: dynamic resource allocationfor mobile multihop multichannel wireless networking* IEEE MILCOM 1999, Vol. 1, November 1999.

[11] V. Rajendran, K. Obraczka, J.J. Garcia-Luna-Aceves, *Energy-efficient, collision-free medium access control for wireless sensor networks*, Sensys'03, Los Angeles, California November 2003.

[12] V. Rajendran, J.J. Garcia-Luna-Aceves, K. Obraczka, *Energy-efficient, application-aware medium access for sensor networks*, IEEE MASS 2005, Washington, November 2005.

[13] I. Rhee, A. Warrier, M. Aia, J. Min, *Z-MAC: a hybrid MAC for wireless sensor networks*, SenSys'05, San Diego, California, November 2005.

[14] I. Rhee, A. Warrier, L. Xu, *Randomized dining philosophers to TDMA scheduling in wireless sensor networks*, Technical Report TR-2005-21, Dept of Computer Science, North Carolina State University, April 2005.

[15] P. Minet, S. Mahfoudh, *SERENA: SchEduling RoutEr Nodes Activity in wireless ad hoc and sensor networks*, IWCMC 2008, IEEE International Wireless Communications and Mobile Computing Conference, Crete Island, Greece, August 2008.

[16] D. Brelaz, *New methods to color the vertices of a graph*, Communications of the ACM, 22(4), 1979.

[17] J. Peemoller, *A correction to Brelaz's modification of Brown's coloring algorithm*, Communications of the ACM, 26(8), 1983.

[18] M. Garey, D. Johnson, *Computers and intractability: a guide to theory of NP-completeness*, W.H. Freeman, San Francisco, California, 1979.

[19] I. Finoccho, A. Panconesi, R. Silvestri, *Experimental analysis of simple distributed vertex coloring algorithms*, SODA 2002, San francisco, California, January 2002.

[20] C. Busch, M. Magdon-Ismail, F. Sivrikaya, B. Yener, *Contention-free MAC protocols for wireless sensor networks*, DISC 2004, Amsterdam, Netherlands, October 2004.

[21] F. Kuhn, R. Watttenhofer, *On the complexity of distributed graph coloring*, PODC 2006, Denver, Colorado, USA, July 2006.

[22] J. Hansen, M. Kubale, L. Kuszner, A. Nadolski, *Distributed largest-first algorithm for graph coloring*, EURO-PAR 2004, Pisa, Italy, August 2004.

[23] S. Gobriel, R. Cleric, D. Mosse, *Adaptations of TDMA scheduling for wireless sensor networks*, RTN 2008, International Workshop on Real-Time Networks, Prague, Czech Republic, July 2008.

[24] T. Herman, S. Tixeuil, *A distributed TDMA slot assignment algorithm for wireless sensor networks*, ALGOSENSORS 2004, Turku, Finland, July 2004.

[25] OCARI project, http://ocari.lri.fr/.